**Use Cases - Strings**

The Redis string data structure is the most versatile data structures that can be used across multiple use cases;

* **For serving static websites pages.** Redis.io site uses strings to serve all static page contents
* **Caching** - To store most common, frequently used data within an application or a website.
* **Counters** are excellent choice for strings e.g Daily website visitors and more.
* **Master Catalog and configuration** e.g. You can store all the application default or user based configuration settings in respective strings key e.g.  
    
  SET app:config:website www.test.com
* The Redis String type is the simplest type of value you can associate with a Redis key.
* String is not the best term for this type because it is used to hold numeric types as well.
* Internally it is held as a byte array.
* A string is a simple scalar that can hold a
  + single value or
  + it can hold an XML or
  + a JSON document as well
* A string value can't be bigger than 512 MB.

**SET, SETNX, MSET, MSETNX**

Set value, Set value if key does not exist, Set multiple values, Set multiple values if none of the keys exist

**GET, MGET, GETSET**

Get value, Get multiple values, Set value, return old value

**PSETEX, SETEX**

Set with expiry in seconds and milliseconds

**INCR, INCRBY, INCRBYFLOAT**

Increment integer, Add to integer, Add to float

**DECR, DECRBY**

Decrement integer, Subtract from integer

**Use Cases - Lists**

Lists contain strings that are sorted by their insertion order. With Redis Lists, you can add items to the head or tail of the lists, which is very useful for queueing jobs. Some of use cases for the list may be;

1. **Social Networking Sites:**Social platforms like Twitter use Redis Lists to populate their timelines or homepage feeds, and can customize the top of their feeds with trending tweets or stories.
2. **RSS Feeds:**We can create news feeds from custom sources where you can pull the latest updates and allow interested followers to subscribe to their favorite RSS feed.
3. **Leaderboards:** This is more of high use cases where Forums like Reddit and other voting platforms utilize Redis Lists to add articles to the leaderboard and even sort them by most viewed and voted entries.

**Quick Reference - Lists**

* A list is just a sequence of ordered elements.
* What's the downside? Accessing an element *by index* is very fast in lists implemented with an Array (constant time indexed access) and not so fast in lists implemented by linked lists (where the operation requires an amount of work proportional to the index of the accessed element).
* You can think of list as an array
* List are designed in such a way that adding new elements at the end of a list, is really fast.
* The downside is that indexing into the list can be slow.
* When this is required, Sorted Sets are a better option

**LPUSH, RPUSH**

Add Value at beginning, Add value at the end

**LPUSHX, RPUSHX**

Only push if key already exists

**LLEN, LRANGE**

Get number of values, Get values from Start to Stop

**LINDEX, LSET, LINSERT**

Get a value by index, Set a value by index, Add a value before or after another

**LREM, LTRIM**

Delete element from list, Trim list by range

**LPOP, RPOP**

Delete and Get the first element, Delete and Get the last element

**Use Cases - Hashes**

Redis Hashes are maps between string fields and string values. They are the go-to data type if you need to essentially create a container of unique fields and their values to represent objects.

1. **User Profiles:** Redis Hash data structures can be used to store application objects like Users information and more. Many web applications use Redis Hashes for their user profiles, as they can use a single hash for all the user fields, such as name, surname, email, password, etc.e.g. you can use  
     
   **HSET** user:101 name "Joe" age 30 country "USA"
2. **User Posts:** Some of the popular social platforms like instagram uses Redis Hashes for various purposes e.g.
   1. To map all the archived user photos or posts back belong to  a single user. The Redis Hashes hashing mechanism allows them to look up and return values very fast, fit the data needed in memory, and leverage data persistence in the event one of their servers dies.
3. **Storing Multi-Tenant Metrics:**Multi-tenant metrics can utilize Redis hashes to store critical informations e.g They can use hash data structures to record and store their product and sales metrics in a way that guarantees solid separation between each tenant, as hashes can be encoded efficiently in a very small memory space.

You should use Redis Hashes whenever possible, as you can use a small Redis instance to store millions of objects.

**Quick Reference - Hashes**

* hashes are useful for representing objects
* Hashes contain one or more fields.

**HSET, HSETNX, HMSET**

Set field value, Set field value if field does not exist, Set multiple field values

**HGET, HMGET**

Get field value, Get multiple field values

**HLEN, HKEYS, HVALS, HGETALL**

Get Number of fields, Get all field keys, Get all field values, Get all fields and values

**HEXISTS, HDEL**

Check field exists, delete field

**HINCRBY, HINCRBYFLOAT**

Increment field integer value, Increment field float value